Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jurij Šteblaj

Metoda razvrščanja z združevanjem najbližjih sosedov v programu Orange

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preglejte literaturo na področju gradnje filogenetskih dreves z uporabe tehnike združevanja najbližjih sosedov. Postopek združevanja implementirajte v programu Orange. Preučite postopke vizualizacije rezultatov združevanja in izberite najbolj primerne ter te implementirajte v gradniku z grafičnim uporabniškim vmesnikom. Implementacijo uporabite na primerih in komentirajte primernost njene uporabe za podatkovno rudarjenje.

Kazalo

Povzetek

Metoda razvrščanja z združevanjem najbližjih sosedov gradi filogenetska drevesa iz matrike razdalj med objekti. Uporablja se v bioinformatiki za napovedovanje evolucijskih razmerij med biološkimi vrstami. Kljub uporabnosti na širšem področju aplikacij pa ta metoda le redkokdaj zaide med programe za odkrivanje znanj iz podatkov. V sklopu diplomske naloge smo zato to metodo implementirali v obliki gradnika v programu Orange. Razvili smo tudi nekaj metod vizualizacije zgrajenih filogenetskih dreves in te preizkusili na klasičnih podatkih s področja podatkovnega rudarjenja. Rezultati pričajo o uporabnosti metode izven področja bioinformatike.

Ključne besede: strojno učenje, združevanje najbližjih sosedov, bioinformatika, Orange.

Abstract

Neighbour joining builds phylogenetic trees from distance matrices. It is mainly used in bioinformatics for inference of evolutional relationships between species and prediction of common ancestors. Despite its usefulness in various applications it is rarely available in data mining programs. For this reason we implemented neighbour joining as a widget in general-purpose data mining suite Orange. We also developed several methods for visualisation of the inferred phylogenetic trees. Using our implementation on several use cases we have demonstrated that neighbour joining and constructed clustering trees are useful in data mining tasks outside the scope of bioinformatics.

Keywords: machine learning, neighbour joining, bioinformatics, Orange.

Poglavje 1

Uvod

Filogenetska drevesa [2] so strukture, ki se uporabljajo v biologiji za predstavitev evolucijskih razmerij med skupinami organizmov. Vozlišča v drevesu predstavljajo organizme, povezave med vozlišči pa povezujejo neposrednega evolucijskega prednika in naslednika. Filogenetska drevesa so bila uporabljena v različnih primerih, kot sta analiza širjenja ebole [3] in taksonomija plazilcev [19]. Primer filogenetskih dreves prikazuje slika 1.1.

Ena izmed metod za gradnjo filogenetskih dreves je metoda razvrščanja z združevanjem najbližjih sosedov [13]. Metoda zgradi filogenetsko drevo iz matrike razdalj med organizmi. Uporabljene razdalje pogosto temeljijo na genetskih razlikah med organizmi, lahko pa so izračunane tudi iz drugih podatkov. Filogenetska drevesa lahko vizualiziramo z različnimi tehnikami, kar vpliva na preglednost vizualizacije. Obstaja veliko orodij za vizualizacijo filogenetskih dreves, kot sta Dendroscope[1] in IcyTree[2].

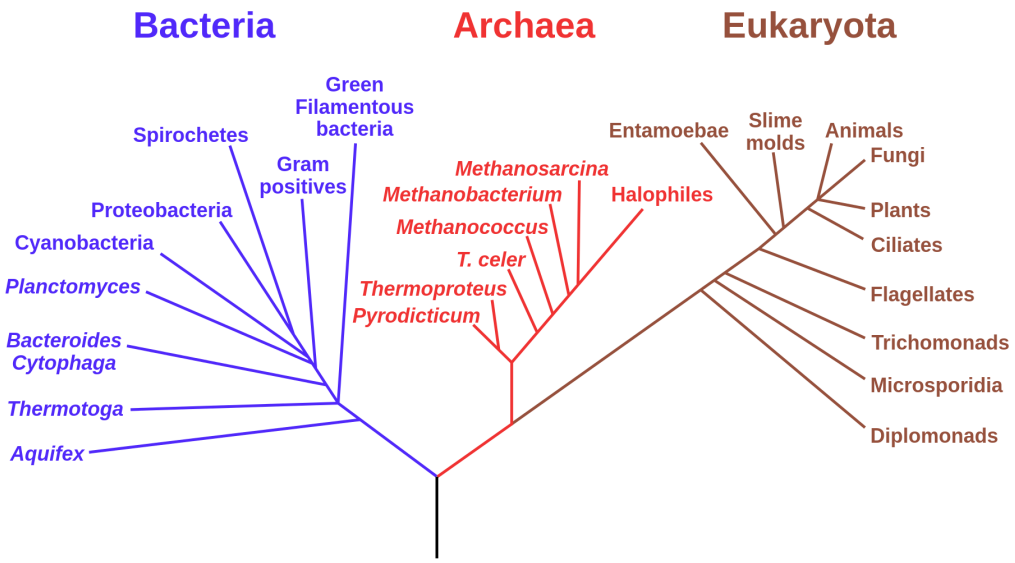
Program Orange[3], ki ga razvija Laboratorij za bioinformatiko Fakultete za računalništvo in informatiko, je orodje, ki se uporablja za odkrivanje znanj iz podatkov [6]. Orange podpira vizualno programiranje, kjer uporabnik povezuje gradnike, ki obdelujejo podatke [4]. Trenutno Orange še nima na voljo metode razvrščanja z združevanjem najbližjih sosedov, zato smo jo v sklopu

[1] `http://ab.inf.uni-tuebingen.de/software/dendroscope/`

[2] `https://icytree.org/`

[3] `https://orange.biolab.si/`

diplomske naloge implementirali kot nov gradnik. Razvili smo pripadajoči del knjižnice za gradnjo ter vizualizacijo zgrajenih dreves.



Slika 1.1: Filogenetsko drevo. Vir: Wikipedija[4].

[4] https://en.wikipedia.org/wiki/Phylogenetic_tree

Poglavje 2

Razvrščanje z združevanjem sosedov

Metodo razvrščanja z združevanjem najbližjih sosedov sta prva predlagala Saitou in Nei [13]. Spada med metode aglomerativnega gručenja in izdela filogenetsko drevo na podlagi matrike razdalj. Če so točke v matriki razdalj v metričnem prostoru, bo metoda razvrščanja z združevanjem sosedov zgradila aditivno drevo, kjer bo pot med dvema točkama po vejah drevesa enako dolga kot razdalja med tema dvema točkama v matriki. Za vse trojke točk x, y, z v metričnem prostoru velja:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \iff x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$

Alternativen zapis metode, ki ima časovno kompleksnost $O(n^3)$, sta razvila Studier in Keppler [18]. Postopek si lahko predstavljamo, kot da imajo najprej vsa vozlišča, torej vse točke oziroma objekti, med katerimi smo izmerili razdalje, skupnega očeta. Nato jih paroma združujemo in dodajamo nove vmesne očete, ki so očetje združenih dveh vozlišč in sinovi prejšnjih očetov

združenih vozlišč. Metoda razvrščanja z združevanjem najbližjih sosedov je tako požrešen algoritem, ki minimizira dolžino drevesa, kjer je dolžina določena s posplošeno Pauplinovo formulo $l = \sum_{\{i,j\}} w(i,j)d(i,j)$. Tu je $d(i,j)$ razdalja v drevesu po vejah med listoma i in j in $w(i,j) = \frac{1}{\prod_k c(k)}$, kjer so k vozlišča na poti med listoma i in j, $c(k)$ pa je število izhodnih povezav takega vozlišča [7]. Pauplinovo formulo za dolžino drevesa je prvič za binarna drevesa zapisal Pauplin leta 2000 [12], leta 2004 pa sta jo posplošila Semple in Steel [15].

2.1 Postopek združevanja sosedov

Metodo razvrščanja z združevanjem najbližjih sosedov lahko zapišemo na sledeč način:

1. Izberemo dve vozlišči i, j, ki minimizirata

$$Q(i,j) = (n-2)D(i,j) - \sum_{k=1}^{n} D(i,k) - \sum_{k=1}^{n} D(j,k), \tag{2.1}$$

   kjer je n število preostalih vozlišč v matriki razdalj, $D(x,y)$ pa razdalja v matriki razdalj med vozliščema x in y.

2. Razdalji v drevesu od izbranih vozlišč i, j do novega vozlišča u sta

$$\delta(i,u) = \frac{1}{2}D(i,j) + \frac{1}{2(n-2)}\left(\sum_{k=1}^{n} D(i,k) - \sum_{k=1}^{n} D(j,k)\right) \tag{2.2}$$

   in

$$\delta(j,u) = \frac{1}{2}D(i,j) + \frac{1}{2(n-2)}\left(\sum_{k=1}^{n} D(j,k) - \sum_{k=1}^{n} D(i,k)\right). \tag{2.3}$$

3. Izračunamo razdalje med novim vozliščem u in preostalimi vozlišči k

$$D(k,u) = \frac{1}{2}(D(i,k) - \delta(i,u)) + \frac{1}{2}(D(j,k) - \delta(j,u)). \tag{2.4}$$

4. Iz matrike D odstranimo vozlišči i, j in dodamo vozlišče u. Če sta preostali več kot dve vozlišči, se vrnemo na 1. korak.

5. Povežemo preostali dve vozlišči x in y. Razdalja med njima v drevesu je enaka razdalji med njima v matriki:

$$\delta(x, y) = D(x, y) \tag{2.5}$$

Oglejmo si primer uporabe tega postopka na spodnji matriki razdalj:

$$D_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 3 & 7 & 6 \\ 3 & 0 & 6 & 5 \\ 7 & 6 & 0 & 5 \\ 6 & 5 & 5 & 0 \end{pmatrix} \end{array}$$

Po enačbi (2.1) izračunamo vrednosti Q za vsak par vozlišč i, j. Za vozlišči a in b je ta vrednost podana s spodnjim izrazom:

$$Q_1(a, b) = (4 - 2)3 - (0 + 3 + 7 + 6) - (3 + 0 + 6 + 5) = -24$$

Celotna matrika Q_1 pa je:

$$Q_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} & -24 & -20 & -20 \\ -24 & & -20 & -20 \\ -20 & -20 & & -24 \\ -20 & -20 & -24 & \end{pmatrix} \end{array}$$

Izberemo lahko katerikoli par i, j, kjer je $Q_1(i, j)$ najmanjši (vrednost -24). Tukaj izberemo vozlišči a in b in ju združimo v vozlišče e. Nato skladno z enačbami (2.2) in (2.3) izračunamo razdalji v drevesu od izbranih vozlišč a in b do novega vozlišča e.

$$\delta(a, e) = \frac{1}{2}3 + \frac{1}{2(4-2)}((0 + 3 + 7 + 6) - (3 + 0 + 6 + 5)) = 2$$
$$\delta(b, e) = \frac{1}{2}3 + \frac{1}{2(4-2)}((3 + 0 + 6 + 5) - (0 + 3 + 7 + 6)) = 1$$

Po enačbi (2.4) izračunamo razdaljo med preostalima vozliščema c, d in novim vozliščem e.

$$D_2(c,e) = \frac{1}{2}(7-2) + \frac{1}{2}(6-1) = 5$$
$$D_2(d,e) = \frac{1}{2}(6-2) + \frac{1}{2}(5-1) = 4$$

Nova matrika razdalj, ki namesto vozlišč a, b vsebuje njunega prednika e je:

$$D_2 = \begin{array}{c} \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccc} c & d & e \end{array} \\ \begin{pmatrix} 0 & 5 & 5 \\ 5 & 0 & 4 \\ 5 & 4 & 0 \end{pmatrix} \end{array}$$

Ker sta preostali več kot dve vozlišči, začnemo naslednjo iteracijo algoritma, ter izračunamo matriko Q_2:

$$Q_2 = \begin{array}{c} \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccc} c & d & e \end{array} \\ \begin{pmatrix} & -14 & -14 \\ -14 & & -14 \\ -14 & -14 & \end{pmatrix} \end{array}$$

Izberemo lahko katerikoli par i, j, kjer je $Q_2(i,j)$ najmanjši (vrednost -14). Tukaj izberemo vozlišči c in d in ju združimo v vozlišče f. Razdalji od vozlišč c in d do novega vozlišča sta:

$$\delta(c,f) = 3$$
$$\delta(d,f) = 2$$

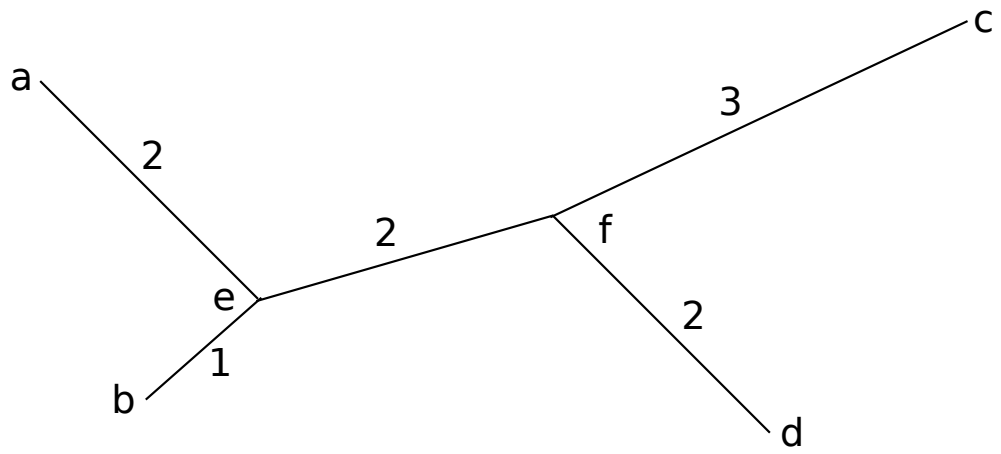
Matrika razdalj med preostalimi vozlišči pa je:

$$D_3 = \begin{array}{c} \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cc} e & f \end{array} \\ \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix} \end{array}$$

Ker imamo le še dve vozlišči, ju združimo. Razdalja v drevesu je enaka razdalji v matriki razdalj (2.5).

$\delta(e, f) = 2$

Rezultat postopka je drevo, ki ga lahko ponazorimo z grafom (slika 2.1).



Slika 2.1: Drevo, ki nastane z metodo razvrščanja z združevanjem najbližjih sosedov za matriko razdalj D_1.

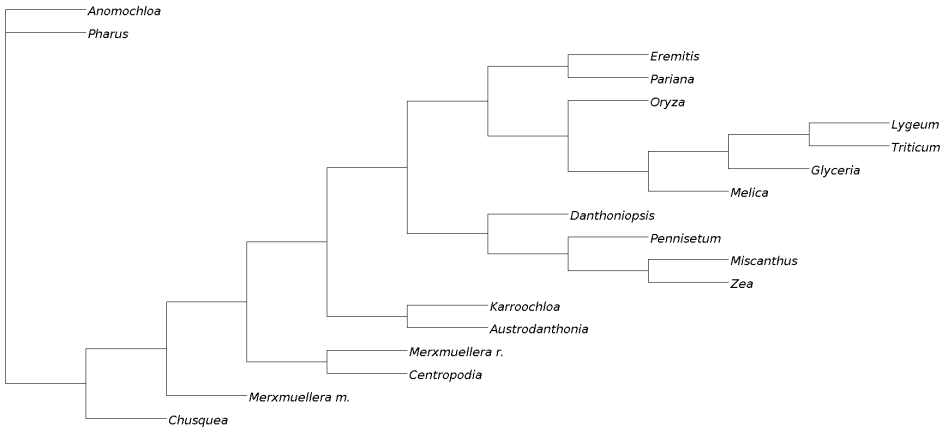
Razdalje v drevesu, zgrajenim z metodo razvrščanja z združevanjem najbližjih sosedov, so lahko tudi negativne. Negativne razdalje med točkami nimajo dobre biološke razlage, poleg tega pa zelo slabo vplivajo na preglednost vizualizacij. Zaradi tega jih pri naši implementaciji odstranimo s funkcijo v poglavju 3.2. Ta postopek zagotovi, da so vse razdalje med točkami v drevesu nad izbrano vrednostjo.

2.2 Vizualizacije filogenetskih dreves

Filogenetsko drevo, ki je rezultat metode razvrščanja z združevanjem najbližjih sosedov, je možno predstaviti z različnimi tehnikami vizualizacij. Vizualizacija pretvori podatke o povezavah med točkami in dolžinah teh povezav v seznam koordinat, kjer bodo točke prikazane. Dobra vizualizacija

zadosti kriterijem, kot so vernost razdalj med točkami, preglednost posameznih točk, preglednost povezav med točkami.

Tehnika, ki je že uporabljena v programu Orange za hierarhično gručenje, so dendrogrami, katerih primer je viden na sliki 2.2. Ena izmed slabosti te tehnike pri vizualizaciji filogenetskih dreves, sestavljenih z razvrščanjem z združevanjem najbližjih sosedov, je, da zaradi usmerjenosti drevo izgleda, kot da ima določen koren. Metoda razvrščanja z združevanjem najbližjih sosedov izdela drevo, ki nima določenega korena, zato mora biti koren izbran ločeno od algoritma. To lahko povzroči, da drevo na vizualizaciji izgleda neuravnoteženo. Dendrogrami imajo tudi druge slabosti, kot je ta, da je pri velikem številu primerov posamezne liste in oznake težko najti, saj so lahko skriti med daljšimi vejami. To naredi posamezne točke manj pregledne.



Slika 2.2: Dendrogram. Graf je izrisan z orodjem Dendroscope.

Ena izmed tehnik, ki je bila razvita posebej za vizualizacijo filogenetskih dreves, je radialna vizualizacija [1], katere primer prikazuje slika 2.3. Vsakemu listu v tej vizualizaciji je namenjen enak del polnega kota. Tako je vsakemu poddrevesu namenjen kot, ki je premo sorazmeren številu listov v poddrevesu. Pomembno je poudariti, da tudi ta tehnika potrebuje drevo s korenom. Izbira korena vpliva na vizualizacijo, a nima tako močnega vpliva na

končno sliko kot pri dendrogramu. Tehnika je opisana s sledečo psevdokodo, ki je povzeta iz članka [1]:

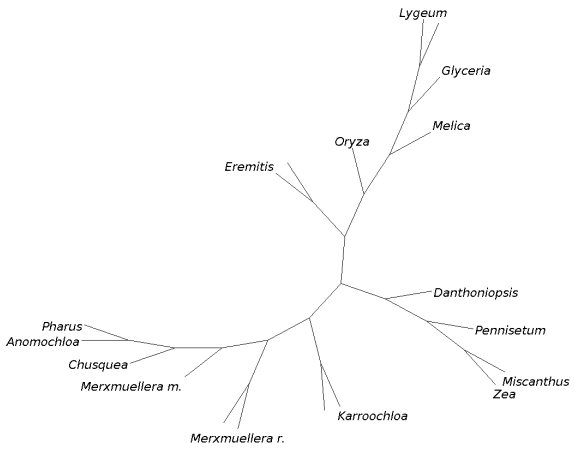
```
begin
    postorder_traversal(root(T))
    x_root(T) ← (0,0)
    ω_root(T) ← 2 * π
    τ_root(T) ← 0
    preorder_traversal(root(T))
end

procedure postorder_traversal(vertex v)
    // Obratno obhodi drevo in vsakemu vozlišču preštej
    // število listov v njegovem poddrevesu l.

    if deg(v) = 1 then
        l_v ← 1
    else
        l_v ← 0
        foreach w ∈ children(v) do
            postorder_traversal(w)
            l_v ← l_v + l_w

procedure preorder_traversal(vertex v)
    // Premo obhodi drevo in vsakemu vozlišču izračunaj
    // koordinate x glede na število listov v njegovem
    // poddrevesu l.

    if v ≠ root(T) then
        u ← parent(v)
        x_v ← x_u + δ(u,v) * (cos(τ_v + ω_v/2), sin(τ_v + ω_v/2))
    η ← τ_v
    foreach w ∈ children(v) do
```

$\omega_w \leftarrow l_w / l_{\texttt{root}(T)}$ * 2 * π
$\tau_w \leftarrow \eta$
$\eta \leftarrow \eta$ + ω_w
`preorder_traversal`(w)

Procedura `postorder_traversal` najprej prešteje za vsako vozlišče v, koliko listov je v poddrevesu s korenom v vozlišču v in to shrani kot l_v. Procedura `preorder_traversal` nato vsakemu vozlišču v dodeli kot ω_v po enačbi

$$\omega_v = \frac{2\pi \cdot l_v}{l_{\text{root}(T)}}. \tag{2.6}$$

Procedura kot pozneje uporabi, da lahko izračuna koordinate vozlišča in jih shrani kot x_v. τ_v predstavlja kot, pri katerem se začne prostor vozlišča v. Prostor, ki je dodeljen vozlišču v, se tako razteza med kotoma τ_v in $\tau_v + \omega_v$. Vozlišče v leži na kotu $\tau_v + \frac{\omega_v}{2}$.



Slika 2.3: Radialna vizualizacija. Graf je izrisan z orodjem Dendroscope.

Druga tehnika, definirana v članku, je krožna tehnika vizualizacije [1], katere primer prikazuje slika 2.4. Ta enakomerno razporedi liste v obliki krožnice, ostala vozlišča pa so znotraj kroga. Položaj vsakega notranjega vozlišča je odvisen od položaja očeta in sinov vozlišča. Tehnika ima to slabost,

da razdalje niso prikazane povsem natančno. Poleg tega je izbira korena bolj pomembna kot pri radialni tehniki, a še vedno ne toliko kot pri dendrogramu. Tehnika ima tudi to prednost, da so listi in njihove oznake, zaradi enakomerne razporeditve listov na enaki razdalji od središča slike, bolj pregledni. Tehnika je opisana s sledečo psevdokodo, ki je povzeta po Bachmaier in sod. [1]:

```
begin
    i ← 0
    k ← 0
    foreach v ∈ V do
        if deg(v) = 1 then k ← k + 1
    postorder_traversal(root(T))
    preorder_traversal(root(T))
end

procedure postorder_traversal(vertex v)
    // Obratno obhodi drevo in vsakemu vozlišču izračunaj
    // koeficient c in odmik d.

    foreach w ∈ children(v) do
        postorder_traversal(w)
    if is_leaf(v) or (v = root(T) and deg(root(T)) = 1) then
        c_v ← 0
        d_v ← (cos(2 * π * i / k), sin(2 * π * i / k))
        i ← i + 1
    else
        S ← 0
        foreach adjacent edge e ← {v, w} do
            if v = root(T) or w = parent(v) then
                s_e ← 1 / δ(e)
            else
                s_e ← 1 / (δ(e) * (deg(v) - 1))
            S ← S + s_e
```

```
        t ← t' ← 0
        foreach outgoing edge e ← (v, w) do
            t ← t + s_e / S * c_w
            t' ← t' + s_e / S * d_w
        if v ≠ root(T) then
            e ← (parent(v), v)
            c_v ← s_e / (S * (1 - t))
        d_v ← t' / (1 - t)

procedure preorder_traversal(vertex v)
    // Premo obhodi drevo in vsakemu vozlišču izračunaj
    // koordinate x glede na koeficient c in odmik d.

    if v = root(T) then
        x_v ← d_v
    else
        u ← parent(v)
        x_v ← c_v * x_u + d_v
    foreach w ∈ children(v) do
        preorder_traversal(w)
```

Namesto dejanskih razdalj se za izračun koordinat točk uporabijo uteži, ki so izračunane po enačbi

$$a_{vw} = \begin{cases} \dfrac{s_{vw}}{\sum_{\{v,k\}\in E} s_{vk}} & \text{če } \{v,w\} \in E, \\ 0 & \text{sicer} \end{cases}, \tag{2.7}$$

kjer s_{vw} izračunamo po enačbi

$$s_{vw} = \begin{cases} \dfrac{1}{\delta(v,w)\cdot(\deg(v)-1)} & \text{če } v = \text{parent}(w), \\ \dfrac{1}{\delta(v,w)} & \text{če } w = \text{parent}(v) \end{cases}. \tag{2.8}$$

Uteži so obratno sorazmerne dolžini povezave. Poleg tega so uteži, ki določajo vpliv sinov na očeta, normirane s številom sinov tega očeta, da številnost

sinov vozlišča z utežmi ne prevlada nad utežjo očeta vozlišča. Procedura `postorder_traversal` za vsako vozlišče v z očetom p izračuna koeficient c_v z enačbo

$$c_v = \begin{cases} 0 & \text{če } v \in \text{leaves}(T) \cup \{\text{root}(T)\}, \\ \dfrac{a_{pv}}{1 - \sum\limits_{(v,w)\in E} (a_{vw} c_w)} & \text{sicer} \end{cases} \tag{2.9}$$

in odmik d_v z enačbo

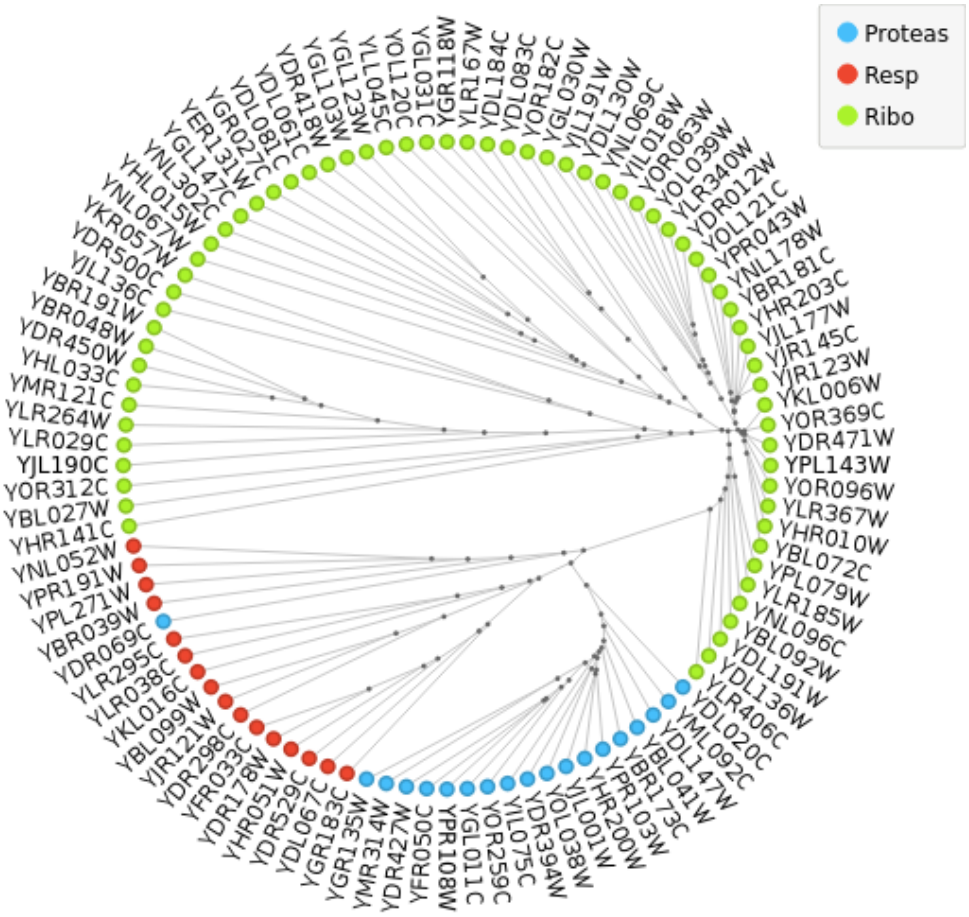
$$d_v = \begin{cases} x_v & \text{če } v \in \text{leaves}(T), \\ \dfrac{\sum\limits_{(v,w)\in E} (a_{vw} d_w)}{1 - \sum\limits_{(v,w)\in E} (a_{vw} c_w)} & \text{sicer} \end{cases} \tag{2.10}$$

glede na uteži na povezavah s sosednjimi vozlišči. Potem druga procedura `preorder_traversal` za vsako vozlišče v, katerega oče je vozlišče p, izračuna koordinate po enačbi

$$x_v = \begin{cases} d_v & \text{če } v = \text{root}(T), \\ c_v x_p + d_v & \text{sicer} \end{cases}. \tag{2.11}$$

2.3 Primerjava s hierarhičnim razvrščanjem v skupine

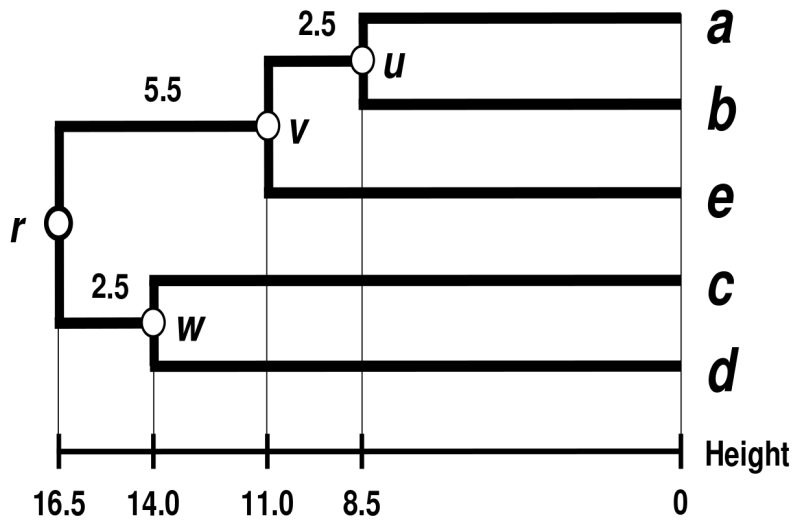
Hierarhično razvrščanje v skupine je že implementirano v programu Orange in je tudi metoda aglomerativnega gručenja. Deluje tako, da je na začetku vsak primer v svoji ločeni gruči. V vsaki iteraciji metoda poišče par gruč, med katerima je razdalja najnižja in ju združi v novo gručo. Razdaljo med dvema gručama lahko izračunamo na različne načine. Pogosto je to povprečje, največja ali najmanjša izmed razdalj med elementi dveh gruč. Postopek ponavljamo, dokler nam ne preostane le ena gruča.



Slika 2.4: Krožna vizualizacija.

Tudi s hierarhičnim razvrščanjem v skupine lahko gradimo filogenetska drevesa. Specifična metoda je UPGMA [17], kjer je razdalja med gručami povprečje razdalj med njihovimi elementi. Ker je pri filogenetskih drevesih pomembno, da lahko ocenimo dolžino poti med pari vozlišč, so razdalje v filogenetskem drevesu UPGMA dvakrat krajše kot v dendrogramu hierarhičnega gručenja, da je lahko vsota oddaljitev sinov od očeta enaka razdalji med obema sinoma [8]. Primer drevesa, ki ga zgradi metoda UPGMA, je viden na sliki 2.5. Čeprav se s hierarhičnim gručenjem da graditi filogenetska

[1] https://en.wikipedia.org/wiki/UPGMA



Slika 2.5: Drevo, ki je nastalo z metodo UPGMA. Vir: Wikipedija[1].

drevesa, ta metoda ne daje tako dobrih rezultatov kot metoda razvrščanja z združevanjem najbližjih sosedov [13, 9]. Ena izmed slabosti je, da hierarhično gručenje predpostavi konstantno hitrost evolucije [11]. To pomeni, da je razdalja od korena do vsakega lista enaka. Zaradi tega ne more zgraditi aditivnega drevesa, ki se ujema z matriko razdalj, v vseh primerih, ko ga lahko metoda razvrščanja z združevanjem sosedov, saj je omejen na gradnjo ultrametričnih dreves [10]. Filogenetsko drevo, ki se povsem ujema z razdaljami v matriki razdalj, lahko zgradi le takrat, kadar so točke v ultrametričnem prostoru. Za vse trojke točk x, y, z v ultrametričnem prostoru velja:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \iff x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq \max\{d(x, y), d(y, z)\}$

Razlika je v četrtem pogoju, ki je za ultrametričen prostor strožji od tistega za metričen prostor [14]. Metoda razvrščanja z združevanjem najbližjih sosedov nima te omejitve in predpostavke in lahko zato izdela boljši približek dejanskega evolucijskega drevesa.

Poglavje 3

Implementacija v sistemu Orange

Metode razvrščanja z združevanjem najbližjih sosedov in tehnike vizualizacije smo implementirali kot gradnik v programu Orange [6]. Implementirali smo radialno in krožno tehniko vizualizacije [1] zaradi lastnosti, ki smo jih že opisali v poglavju 2.2. Kompatibilnost z ostalimi gradniki smo ohranili z uporabo metod in struktur, ki so vključene v programu Orange [5]. Pri implementaciji smo kodo ločili na dva dela: programsko knjižnico in gradnik za program Orange. S knjižnico lahko izvedemo celotno metodo razvrščanja z združevanjem najbližjih sosedov in tako pretvorimo matriko razdalj v drevesno strukturo z razdaljami. Knjižnica vsebuje tudi obe izbrani tehniki vizualizacije, s katerima lahko drevesno strukturo z razdaljami pretvorimo v seznam koordinat, kjer se vozlišča nahajajo na risalni površini. Poleg tega lahko s knjižnico zvišamo razdalje v drevesu na želeno najnižjo vrednost in tako odstranimo negativne vrednosti. Čeprav je bila knjižnica izdelana z namenom, da bo uporabljena v gradniku za program Orange, se jo da uporabljati tudi zunaj tega.

3.1 Programske zahteve

1. Našemu gradniku je najbolj podoben gradnik za hierarhično gručenje (Hierarchical Clustering), saj oba izvajata algoritem gručenja na podlagi razdalj. Zaradi tega ima novi gradnik, tako kot tisti za hierarhično gručenje, vhod, ki sprejme podatke v obliki matrike razdalj.

2. Kot gradnik za hierarhično gručenje, ima tudi novi gradnik dva izhoda. V oba pošlje podatke, iz katerih so bile izračunane razdalje, ne pa samih razdalj. V enega pošlje le podatke v vizualizaciji izbranih primerov, v drugega pa podatke vseh primerov, ampak podatkom doda stolpec, ki pove, kateri primeri so označeni.

3. Novi gradnik prav tako omogoča izbiro izrisanih točk, ki so nastale z izrisom primerov. Izbira deluje enako kot pri gradniku za razsevni diagram (Scatter Plot).

4. Gradnik se mora, tako kot ostali gradniki v programu Orange, takoj odzvati na nove podatke na vhodu. Ker je metoda razvrščanja z združevanjem najbližjih sosedov časovno zahtevna, moramo zagotoviti, da program Orange med delovanjem gradnika ostane odziven. Iz istega razloga uporabniku tudi prikažemo napredek.

5. Grafični vmesnik je skladen z grafičnim vmesnikom ostalih gradnikov, še posebno z grafičnim vmesnikom gradnika za razsevni diagram. Vsebuje enake elemente za nadzor delovanja in ostala orodja, razen tistih, ki bi bili za ta gradnik nesmiselni ali neuporabni.

3.2 Podatkovne strukture

V izdelani knjižnici smo uporabili podatkovne strukture iz programske knjižnice NumPy[1]. Ta vključuje med drugim vektorje in matrike, na katerih se

[1] `http://www.numpy.org/`

operacije izvajajo hitro. Prav so nam prišle tudi matrike, v katerih lahko skrijemo eno ali več vrednosti[2]. Tako skrite vrednosti so izključene iz nekaterih operacij. Funkcija za iskanje minimalne vrednosti v taki matriki na primer preskoči skrite vrednosti. Knjižnico uporablja mnogo programov, med katerimi je tudi Orange [5].

V programu Orange so vključena programska orodja v obliki programske knjižnice. Ta zajema metode in podatkovne strukture. Podatkovne strukture v sistemu Orange omogočajo tvorjenje kompleksnih objektov, ki olajšajo programiranje gradnikov za program Orange. Objekt, ki predstavlja matriko razdalj, lahko tako s sabo med gradniki prenaša povezavo na objekt, ki predstavlja tabelo, iz katere je bila matrika izračunana. Ta objekt, ki predstavlja tabelo, pa lahko vsebuje podatke o paleti, ki se uporablja za barvanje razredov, da so lahko primeri konsistentno obarvani v vseh gradnikih, ki delajo s temi podatki.

Poleg obstoječih struktur smo razvili tudi svoje. Ena izmed takih je struktura drevesa. Spodnja koda v jeziku Python predstavlja drevo s slike 2.1.

```
drevo = {
    "a": [["e", 1]],
    "b": [["e", 2]],
    "c": [["f", 3]],
    "d": [["f", 2]],
    "e": [["a", 1], ["b", 2], ["f", 2]],
    "f": [["c", 3], ["d", 2], ["e", 2]]
}
```

Metoda razvrščanja z združevanjem najbližjih sosedov sestavi drevo brez korena. Ker pa sta bili izbrani tehniki vizualizacije zasnovani za drevesa s korenom, moramo tudi drevesu umetno dodati koren. Prejšnjo strukturo lahko spremenimo tako, da naredimo vse povezave enosmerne. Povezave so

[2] https://docs.scipy.org/doc/numpy/reference/maskedarray.generic.html

potem v smeri od očeta do sina v drevesu. Spodnja koda v jeziku Python predstavlja različico zgornjega drevesa s korenom v vozlišču a.

```
drevo =
    "a": [["e", 1]],
    "b": [],
    "c": [],
    "d": [],
    "e": [["b", 2], ["f", 2]],
    "f": [["c", 3], ["d", 2]]
```

Pomembno je, da iz drevesa odstranimo negativne razdalje. Takšna sprememba rezultata se nam zdi upravičena, saj negativne razdalje nimajo dobre biološke razlage in slabo vplivajo na preglednost vizualizacij. Negativne razdalje odstranimo s spodnjo funkcijo:

```
def set_distance_floor(tree, min_dist):
    for l in tree.values():
        if len(l) == 1 and l[0][1] < min_dist:
            l[0][1] = min_dist
        if len(l) == 2:
            if l[0][1] + l[1][1] < 2 * min_dist:
                l[0][1] = l[1][1] = min_dist
            else:
                for i in range(2):
                    if l[i][1] < min_dist:
                        l[(i + 1) % 2][1] += l[i][1] - min_dist
                        l[i][1] = min_dist
        if len(l) == 3:
            if sum(c[1] for c in l) < 3 * min_dist:
                l[0][1] = l[1][1] = l[2][1] = min_dist
            else:
```

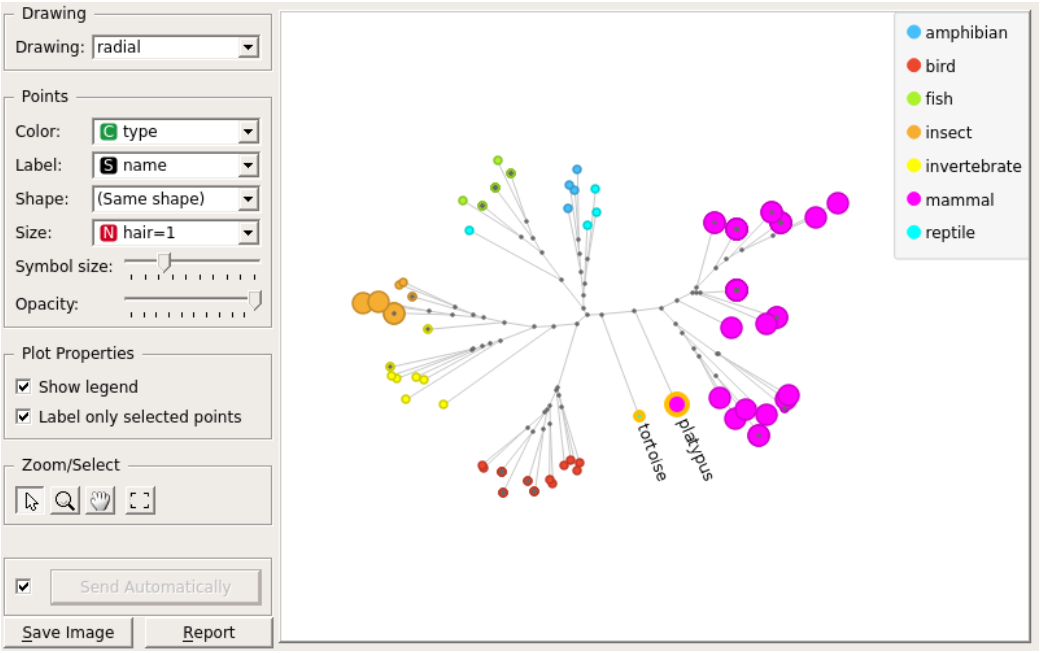
```
for i in range(3):
    if l[i][1] < min_dist:
        l[(i + 1) % 3][1] += l[i][1] - min_dist
        l[i][1] = min_dist
```

Argument `tree` je drevo s korenom, kakršno je opisano v prejšnjem odstavku, argument `min_dist` pa je najnižja vrednost, ki jo želimo dopustiti v drevesu. Funkcija iterira po vozliščih in najprej preveri število izhodnih povezav. Če je povprečje vseh razdalj nižje od `min_dist`, se razdalj ne da tako prerazporediti, da so vse razdalje nad `min_dist` in je vsota razdalj še vedno enaka. Takrat funkcija nastavi razdalje na `min_dist`. Če je povprečje razdalj višje ali enako `min_dist`, funkcija razdalje prerazporedi tako, da ostane vsota razdalj enaka, vse razdalje pa so večje ali enake `min_dist`. Za našo uporabo je najboljši `min_dist`, ki je čim manjši in še vedno pozitiven.

3.3 Grafični vmesnik

Za vključitev knjižnice v grafični del programa Orange smo razvili gradnik z grafičnim vmesnikom, ki ga lahko vidimo na sliki 3.1. Kot pri ostalih gradnikih z izrisom je grafični vmesnik tega gradnika razdeljen na del za nadzor delovanja na levi in del z izrisom na desni strani.

V delu za nadzor delovanja je na vrhu možnost izbire tehnike vizualizacije. Za tem sledijo možnosti izgleda izrisanih točk. Ta del je skladen z delom v gradniku za razsevni diagram. Uporabnik lahko tu nastavi, kako naj se izrišejo točke glede na podatke, ki so bili uporabljeni za izračun matrike razdalj. Od podatkov so lahko odvisne barve, oznake, simboli in velikost simbolov. Uporabnik lahko tudi dodatno poveča ali zmanjša velikost simbolov in lahko nastavi prosojnost simbolov. V naslednjem delu lahko uporabnik dodatno spreminja lastnosti izrisa. Vklopi in izklopi lahko legendo, vklopi pa lahko tudi možnost, da se oznake prikažejo le ob izbranih točkah. To lahko pride prav, ko je veliko točk zelo blizu skupaj, kar se lahko zgodi pri radialni tehniki vizualizacije. Naslednji del vsebuje orodja za premikanje pogleda



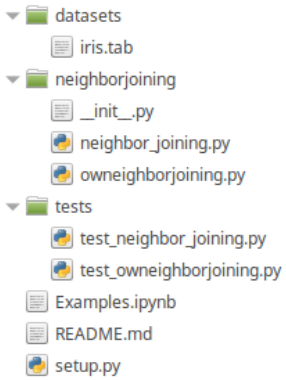
Slika 3.1: Grafični vmesnik gradnika.

na izris in izbiro točk. Prvo omogoča pravokotno izbiro točk in spreminjanje povečave. Drugo omogoča bolj natančno povečavo, saj uporabnik označi pravokotnik, kamor se premakne pogled. Tretje orodje omogoča povečavo, kot jo omogoča prvo orodje, hkrati pa omogoča premikanje pogleda s premikanjem risalne površine. Četrti gumb ponastavi pogled na privzeti pogled. Privzeti pogled je odvisen od postavitve točk in dolžine oznak. Sledita gumb, ki posodobi izbiro primerov na izhodu, in možnost, da se to izvaja samodejno. Na koncu sta še gumb, ki shrani izris kot sliko v samostojno datoteko, in gumb, ki izdela poročilo s sliko in njenim pojasnilom.

Na desni strani grafičnega vmesnika gradnika je izris rezultata metode razvrščanja z združevanjem najbližjih sosedov, izbrane tehnike vizualizacije in ostalih nastavitev iz desnega dela gradnika. Poleg samega izrisa je na voljo tudi legenda, ki razloži barvo in obliko simbolov. Legendo se da premikati po glavnem delu gradnika, lahko pa se jo tudi skrije z nadzornim elementom v delu za nadzor delovanja na levi.

3.4 Skriptna knjižnica

Implementacija je strukturirana kot dodatek za program Orange in je dostopna na repozitoriju GitHub[3]. Dodatek namestimo tako, da ga prenesemo na naš računalnik in znotraj prenesenega repozitorija uporabimo ukaz `pip install .`. Predvidoma se bo koda prenesla v zbirko prototipnih gradnikov[4] in kasneje v sam program Orange. Slika 3.2 prikazuje datoteke v repozitoriju. Datoteka `README.md` vsebuje opis projekta, navodila za namestitev dodatka



Slika 3.2: Datoteke v repozitoriju.

in navodila za uporabo knjižnice. Mapa `neighborjoining` vsebuje datoteko `neighbor_joining.py`, v kateri je implementirana knjižnica, in datoteko `owneighborjoining.py`, v kateri je implementiran gradnik. Mapa `tests` vsebuje avtomatske teste za knjižnico in gradnik. Datoteka `Examples.ipynb` vsebuje primere uporabe knjižnice, mapa `datasets` pa vsebuje podatke, ki jih ti primeri uporabljajo. Datoteka `setup.py` se uporablja pri namestitvi.

[3] `https://github.com/jurijsteblaj/orange-neighbor-joining`
[4] `https://github.com/biolab/orange3-prototypes`

Poglavje 4

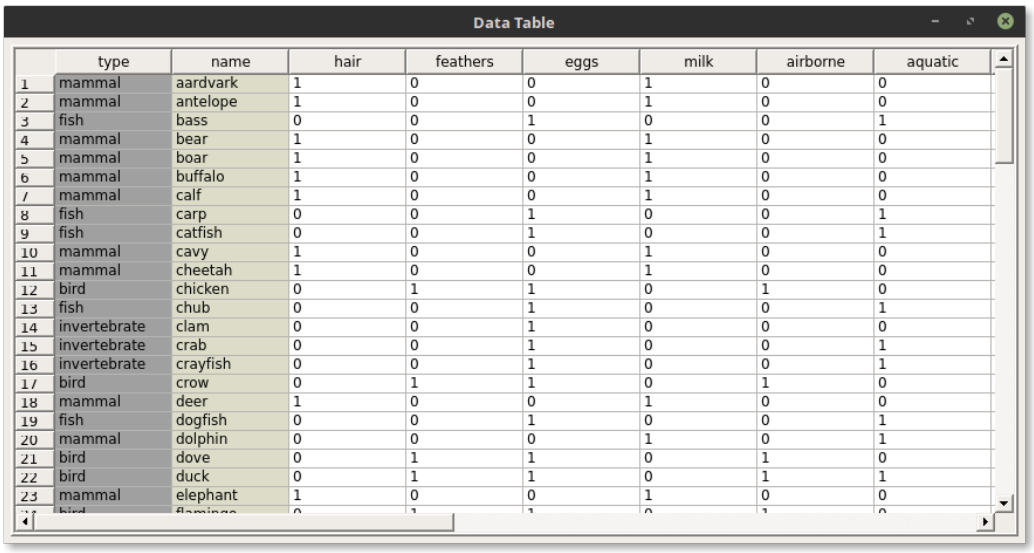
Primeri uporabe

Uporabnost gradnika smo preverili na treh naborih podatkov, ki pa so bili zelo različni. Prvi je imel diskretne atribute, drugi je bil sestavljen iz slik, tretji pa je imel zvezne atribute. Tako smo metodo preizkusili tudi zunaj področja bioinformatike.

4.1 Živalski vrt

Slika 4.1 prikazuje podatke, ki jih bomo uporabili v prvem primeru. Gre za znan nabor podatkov Zoo, ki je na voljo v UCI zbirki podatkov za strojno učenje[1]. Podatki imajo 16 atributov, po katerih bomo razvrščali, in 101 primer. Atributi v teh podatkih so opisni, kot je število nog, ali ima žival dlako in ali je žival udomačena. Lahko si predstavljamo, da naletimo na do sedaj še neznano vrsto in jo skušamo uvrstiti glede na lastnosti, ki jih lahko takoj opazimo. Slika 4.2 prikazuje primer uporabe metode na teh podatkih. Podatke najprej naložimo iz datoteke (gradnik File). Nato moramo diskretne podatke pretvoriti v zvezne (gradnik Continuize). Iz zveznih podatkov nato izračunamo matriko razdalj (gradnik Distances). Z novim gradnikom iz razdalj izdelamo filogenetsko drevo in ga vizualiziramo. Krožna tehnika vizualizacije omogoča boljšo preglednost nad vsemi točkami in njihovimi oznakami

[1] `https://archive.ics.uci.edu/ml/datasets/Zoo`

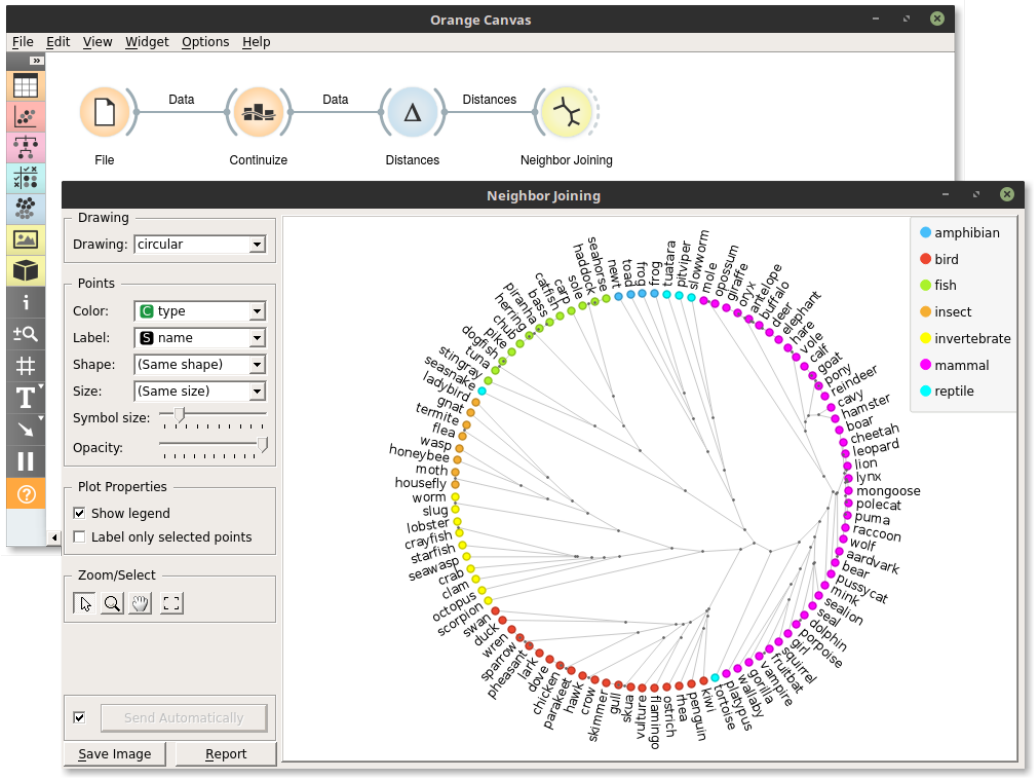


	type	name	hair	feathers	eggs	milk	airborne	aquatic
1	mammal	aardvark	1	0	0	1	0	0
2	mammal	antelope	1	0	0	1	0	0
3	fish	bass	0	0	1	0	0	1
4	mammal	bear	1	0	0	1	0	0
5	mammal	boar	1	0	0	1	0	0
6	mammal	buffalo	1	0	0	1	0	0
7	mammal	calf	1	0	0	1	0	0
8	fish	carp	0	0	1	0	0	1
9	fish	catfish	0	0	1	0	0	1
10	mammal	cavy	1	0	0	1	0	0
11	mammal	cheetah	1	0	0	1	0	0
12	bird	chicken	0	1	1	0	1	0
13	fish	chub	0	0	1	0	0	1
14	invertebrate	clam	0	0	1	0	0	0
15	invertebrate	crab	0	0	1	0	0	1
16	invertebrate	crayfish	0	0	1	0	0	1
17	bird	crow	0	1	1	0	1	0
18	mammal	deer	1	0	0	1	0	0
19	fish	dogfish	0	0	1	0	0	1
20	mammal	dolphin	0	0	0	1	0	1
21	bird	dove	0	1	1	0	1	0
22	bird	duck	0	1	1	0	1	1
23	mammal	elephant	1	0	0	1	0	0

Slika 4.1: Podatki o živalih. Vsaka od vrstic atributno opiše značilnosti dane živali.

pri večjem številu točk. Vidimo lahko, da metoda pogosto napove skupne prednike živalim, ki spadajo v isto skupino živali.

Podatke lahko še naprej analiziramo, kot lahko vidimo na sliki 4.3. Če nas bolj podrobno zanima zakaj je metoda napovedala, da so si sorodni delfin, pliskavica, tjulenj in morski lev, jih lahko izberemo. Z gradnikom lahko nato odstranimo vse atribute, ki imajo pri izbranih primerih enake vrednosti (gradnik Purge Domain). To zmanjšano tabelo lahko nato prikažemo (gradnik Data Table). Vidimo lahko, da naše vrste razlikuje le posest dlak, število nog in posest repa. Paroma se vrste razlikujejo v največ dveh lastnostih.



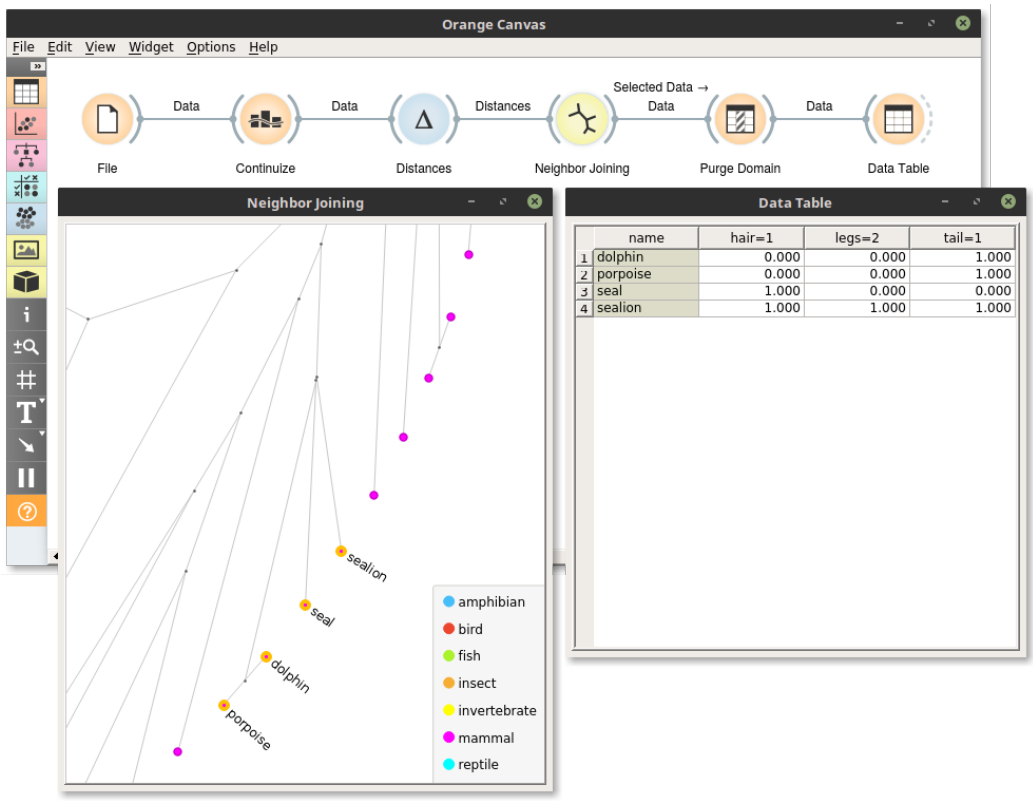
Slika 4.2: Vizualizacija razvrščanja z združevanjem najbližjih sosedov glede na razdalje, izračunane na podlagi lastnosti živali.

4.2 Slike živali

V tem primeru bomo analizirali risbe 19-ih živali[2], ki so bile uporabljene v primeru na spletni strani programa Orange[3]. Vhodne podatke lahko vidimo na sliki 4.4. Lahko si predstavljamo, da naletimo na do zdaj še neznano vrsto in jo skušamo na podlagi fotografije ali risbe uvrstiti v filogenetsko drevo. Slika 4.5 prikazuje primer uporabe metode na teh slikah. Slike uvozimo (gradnik Import Images) in jih pretvorimo v številske vektorje (gradnik Image Embedding). Med temi vektorji izračunamo razdalje (gradnik Distances). Z

[2] https://github.com/ajdapretnar/datasets/blob/master/images/domestic-animals.zip

[3] https://blog.biolab.si/2017/04/03/image-analytics-clustering/

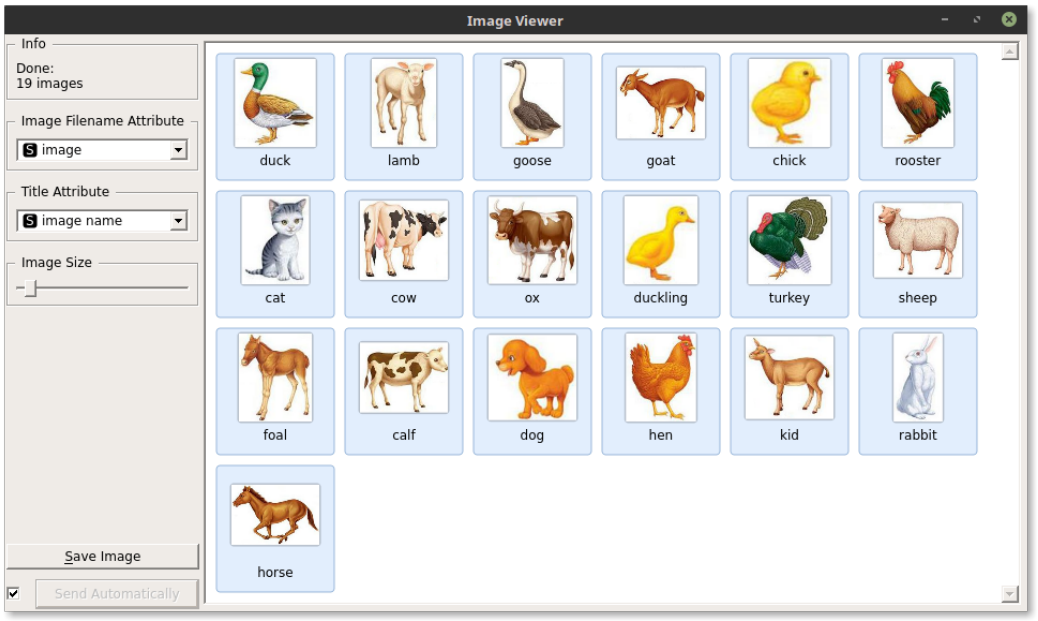


Slika 4.3: Primerjava delfina, pliskavice, tjulnja in morskega leva.

novim gradnikom iz razdalj izdelamo filogenetsko drevo in ga vizualiziramo. Radialna tehnika vizualizacije bolj pregledno prikaže povezave med točkami. Vidimo lahko, kako algoritem napove skupne prednike živalim na slikah. Metoda najprej združi živali iste vrste, kot sta petelin in kokoš. Nato je tema dvema živalma pridružil podobnega purana. Zanimiva napaka je ta, da ta postopek napoveduje, da je piščanec bližje v sorodu z račjim mladičem kot s kokošjo ali petelinom.

4.3 Lastnosti vina

Slika 4.6 prikazuje podatke, ki jih bomo uporabili v prvem primeru. Kot pri prvem primeru gre tudi tu za znan nabor podatkov. Tokrat bomo delali na

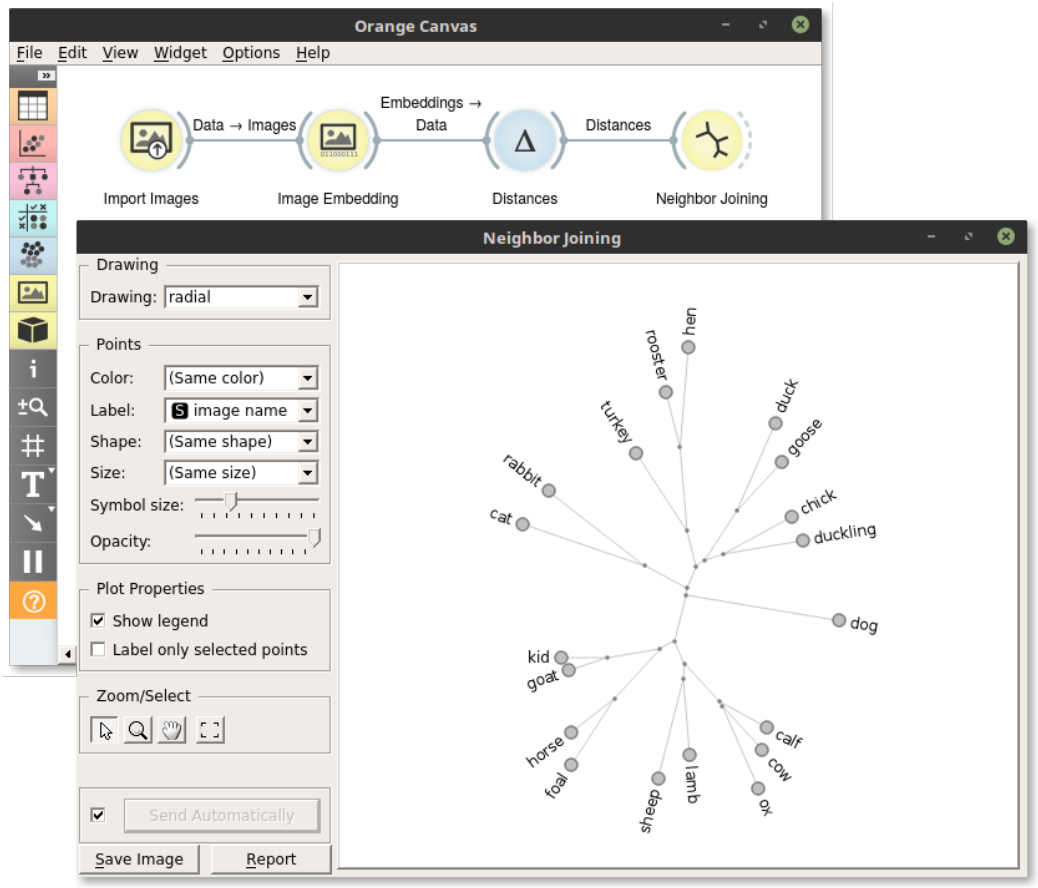


Slika 4.4: Slike živali.

naboru podatkov Wine, ki je tudi na voljo v UCI zbirki podatkov za strojno učenje[4]. Slika 4.7 prikazuje primer uporabe metode na podatkih o vinih. Podatki imajo 13 atributov, po katerih bomo razvrščali, in 178 primerov. Atributi so zvezne lastnosti in količine, kot so odtenek, intenzivnost barve in delež alkohola. Vsako izmed vin pripada enemu izmed treh razredov. Predstavljamo si lahko, da skušamo ugotoviti, kako je potekal razvoj novih vin. Najprej uvozimo podatke (gradnik File). Nato med vini izračunamo razdalje (gradnik Distances). Z novim gradnikom iz razdalj izdelamo filogenetsko drevo in ga vizualiziramo.

Metoda razvrščanja z združevanjem najbližjih sosedov tu ne da dobrih rezultatov. Vizualizacija je nepregledna in ne omogoča, da bi z njeno pomočjo odkrili kaj novega o podatkih. Razlog za nepregleden izris so različni razponi vrednosti atributov. Te povzročijo zelo različne razdalje med primeri, nekatere so tudi zelo blizu ničle. Rezultat lahko izboljšamo s predobdelavo

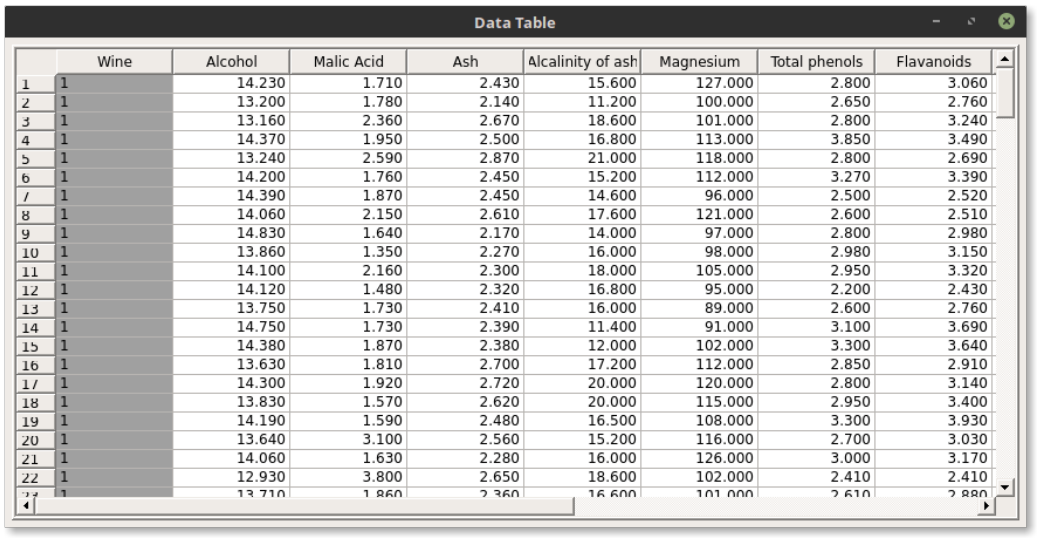
[4] `https://archive.ics.uci.edu/ml/datasets/wine`



Slika 4.5: Vizualizacija razvrščanja z združevanjem najbližjih sosedov glede na razdalje med slikami.

podatkov, kot lahko vidimo na sliki 4.8. To naredimo tako, da med gradnikom, ki prebere podatke iz datoteke, in gradnikom za izračun razdalj podatke normaliziramo (gradnik Continuize).

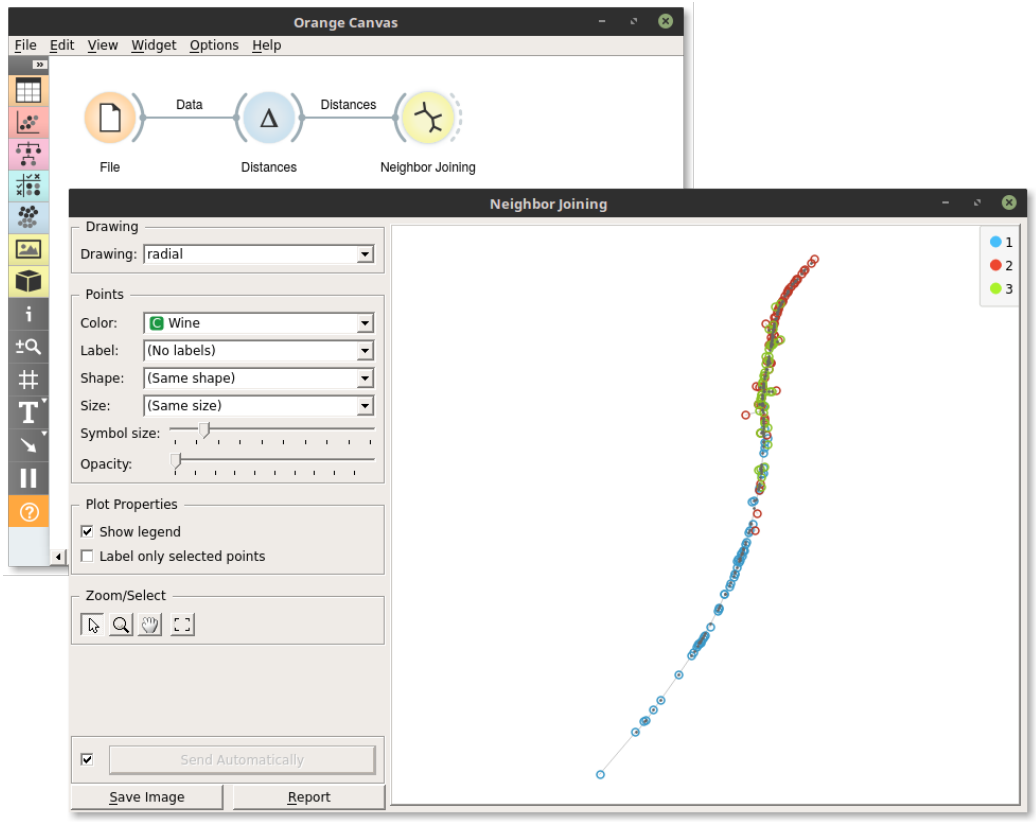
Vseeno pa se moramo vprašati o smiselnosti tega filogenetskega drevesa. Čeprav je metoda razvrščanja z združevanjem najbližjih sosedov lahko še vedno uporabna za gručenje, deluje dobro za izdelavo filogenetskih dreves le, če veljajo nekatere predpostavke. Prva je ta, da so se nasledniki res razvili iz prednikov in so primeri tako bolj podobni svojim prednikom in primerom z bližnjimi skupnimi predniki, kot ostalim primerom. Če bi vsi primeri nastali



	Wine	Alcohol	Malic Acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids
1	1	14.230	1.710	2.430	15.600	127.000	2.800	3.060
2	1	13.200	1.780	2.140	11.200	100.000	2.650	2.760
3	1	13.160	2.360	2.670	18.600	101.000	2.800	3.240
4	1	14.370	1.950	2.500	16.800	113.000	3.850	3.490
5	1	13.240	2.590	2.870	21.000	118.000	2.800	2.690
6	1	14.200	1.760	2.450	15.200	112.000	3.270	3.390
7	1	14.390	1.870	2.450	14.600	96.000	2.500	2.520
8	1	14.060	2.150	2.610	17.600	121.000	2.600	2.510
9	1	14.830	1.640	2.170	14.000	97.000	2.800	2.980
10	1	13.860	1.350	2.270	16.000	98.000	2.980	3.150
11	1	14.100	2.160	2.300	18.000	105.000	2.950	3.320
12	1	14.120	1.480	2.320	16.800	95.000	2.200	2.430
13	1	13.750	1.730	2.410	16.000	89.000	2.600	2.760
14	1	14.750	1.730	2.390	11.400	91.000	3.100	3.690
15	1	14.380	1.870	2.380	12.000	102.000	3.300	3.640
16	1	13.630	1.810	2.700	17.200	112.000	2.850	2.910
17	1	14.300	1.920	2.720	20.000	120.000	2.800	3.140
18	1	13.830	1.570	2.620	20.000	115.000	2.950	3.400
19	1	14.190	1.590	2.480	16.500	108.000	3.300	3.930
20	1	13.640	3.100	2.560	15.200	116.000	2.700	3.030
21	1	14.060	1.630	2.280	16.000	126.000	3.000	3.170
22	1	12.930	3.800	2.650	18.600	102.000	2.410	2.410
23	1	13.710	1.860	2.360	16.600	101.000	2.610	2.880

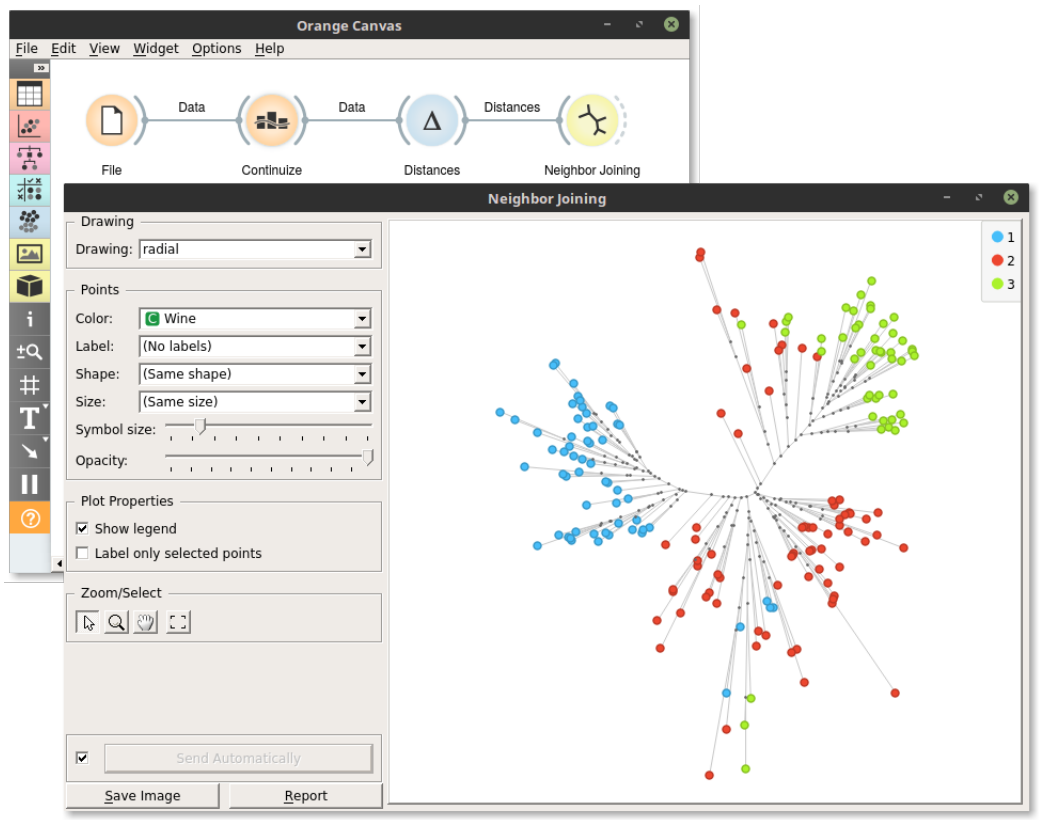
Slika 4.6: Podatki o vinih. Vsaka vrstica z zveznimi vrednostmi opiše posamezno vino.

istočasno ali pa bi bili podatki naključni, to ne bi držalo, saj tako noben par primerov ne bi imel pravega skupnega prednika. Dodatno je z uporabo metode razvrščanja z združevanjem najbližjih sosedov ta predpostavka še strožja, saj predpostavljamo, da imajo vsi primeri skupnega prednika, saj je rezultat eno samo filogenetsko drevo, ki povezuje vse primere. Metoda razvrščanja z združevanjem najbližjih sosedov tako ne bi bila smiselna, če bi razni predniki primerov nastali ločeno brez skupnega prednika. V takem primeru bi bila bolj pravilna ponazoritev z gozdom, kjer bi imela vsaka skupina, kjer imajo primeri skupnega prednika, svoje drevo. V kontekstu vin ne moremo zagotovo reči, ali gre za razvoj, kjer so nasledniki razviti iz prednikov in dedujejo nekatere njihove lastnosti, kar bi naredilo bratske primere podobne. Kljub temu pa lahko vidimo, da je algoritem vseeno uporaben za navadno gručenje, saj je metoda razvrščanja z združevanjem najbližjih sosedov prej povezala primere, ki pripadajo istemu razredu. Podobno velja za poddrevesa v primeru s slikami živali, kjer so v listih le primeri iste vrste. Tako poddrevo nam ne pove veliko in bi ga lahko zamenjali z enim listom ali razredom, nam



Slika 4.7: Vizualizacija razvrščanja z združevanjem najbližjih sosedov glede na razdalje, izračunane na podlagi lastnosti vin.

pa pomaga oceniti pravilnost rezultata. Bolj pravilna filogenetska drevesa bodo v večini primerov imela več poddreves, kjer je visok delež listov istega razreda.



Slika 4.8: Vizualizacija razvrščanja z združevanjem najbližjih sosedov glede na razdalje, izračunane na podlagi normaliziranih lastnosti vin.

Poglavje 5

Zaključek

Metoda razvrščanja z združevanjem najbližjih sosedov se uporablja v bioinformatiki za grajenje filogenetskih dreves. V diplomskem delu smo opisali, kako metoda razvrščanja z združevanjem najbližjih sosedov sestavi filogenetsko drevo iz matrike razdalj. Predstavili smo tudi tehnike vizualizacije filogenetskih dreves. Izdelali smo programsko knjižnico in gradnik za program Orange.

Pokazali smo, da gradnik dobro deluje z razdaljami, ki so bile izračunane na podlagi lastnosti živali. Dobro deluje tudi z razdaljami, ki so bile izračunane na podlagi vektorjev, izpeljanih iz slik. Videli smo tudi, da metoda ne deluje dobro na podatkih z različnimi razponi, zato je včasih podatke potrebno normalizirati.

Implementirali smo najbolj pogosto različico metode razvrščanja z združevanjem najbližjih sosedov, ki pa ima časovno zahtevnost $O(n^3)$. Obstajajo tudi hitrejše implementacije, ki časovno kompleksnost v tipičnih primerih zmanjšajo na $n^2 log(n)$ [16]. Trenutno smo problem z veliko količino podatkov rešili tako, da gradnik avtomatsko vzorči podatke, če je teh preveč. V članku, po katerem smo se zgledovali pri tehnikah vizualizacije, so predlagane tudi nekatere izboljšave [1]. Izboljšana različica radialne tehnike bi lahko nekaterim poddrevesom povečala kote in tako izboljšala preglednost. Izboljšava krožne tehnike bi zmanjšala razlike med dejanskimi in prikazanimi

razdaljami.

Literatura

[1] Christian Bachmaier, Ulrik Brandes, and Barbara Schlieper. Drawing phylogenetic trees. In *International Symposium on Algorithms and Computation*, pages 1110–1121. Springer, 2005.

[2] David Baum. Reading a phylogenetic tree: the meaning of monophyletic groups. *Nature Education*, 1(1):190, 2008.

[3] Miles W Carroll, David A Matthews, Julian A Hiscox, Michael J Elmore, Georgios Pollakis, Andrew Rambaut, Roger Hewson, Isabel García-Dorival, Joseph Akoi Bore, Raymond Koundouno, et al. Temporal and spatial analysis of the 2014-2015 Ebola virus outbreak in West Africa. *Nature*, 524(7563):97–101, 2015.

[4] Tomaz Curk, Janez Demsar, Qikai Xu, Gregor Leban, Uros Petrovic, Ivan Bratko, Gad Shaulsky, and Blaz Zupan. Microarray data mining with visual programming. *Bioinformatics*, 21(3):396–398, 2004.

[5] Janez Demšar, Tomaz Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, et al. Orange: data mining toolbox in Python. *Journal of Machine Learning Research*, 14(1):2349–2353, 2013.

[6] Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From experimental machine learning to interactive data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 537–539. Springer, 2004.

[7] Olivier Gascuel and Mike Steel. Neighbor-joining revealed. *Molecular biology and evolution*, 23(11):1997–2000, 2006.

[8] Paul G Higgs and Teresa K Attwood. *Bioinformatics and molecular evolution*. John Wiley & Sons, 2013.

[9] Thomas Leitner, David Escanilla, Christer Franzen, Mathias Uhlen, and Jan Albert. Accurate reconstruction of a known HIV-1 transmission history by phylogenetic tree analysis. *Proceedings of the National Academy of Sciences*, 93(20):10864–10869, 1996.

[10] Glenn W Milligan. Ultrametric hierarchical clustering algorithms. *Psychometrika*, 44(3):343–346, 1979.

[11] Masatoshi Nei and Li Jin. Variances of the average numbers of nucleotide substitutions within and between populations. *Molecular Biology and Evolution*, 6(3):290–300, 1989.

[12] Yves Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51(1):41–47, 2000.

[13] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[14] Shmuel Sattath and Amos Tversky. Additive similarity trees. *Psychometrika*, 42(3):319–345, 1977.

[15] Charles Semple and Mike Steel. Cyclic permutations and evolutionary trees. *Advances in Applied Mathematics*, 32(4):669–680, 2004.

[16] Luke Sheneman, Jason Evans, and James A Foster. Clearcut: a fast implementation of relaxed neighbor joining. *Bioinformatics*, 22(22):2823–2824, 2006.

[17] Robert R Sokal. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 38:1409–1438, 1958.

[18] James A Studier, Karl J Keppler, et al. A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular biology and evolution*, 5(6):729–731, 1988.

[19] John J Wiens, Caitlin A Kuczynski, Ted Townsend, Tod W Reeder, Daniel G Mulcahy, and Jack W Sites Jr. Combining phylogenomics and fossils in higher-level squamate reptile phylogeny: molecular data change the placement of fossil taxa. *Systematic Biology*, 59(6):674–688, 2010.